

Beadandó feladateleírás – Python

Eszterházy Károly Katolikus Egyetem

Informatikai Kar – Multi paradigmás programozási nyelvek gyakorlat

Kovács Ádám – egyetemi tanársegéd

Célkitűzés

A beadandó célja egy **mikroszerviz-szerű Python-rendszer** megtervezése és megvalósítása, amely bemutatja a hallgató gyakorlati ismereteit a **modern szoftverfejlesztés**, a **programozási paradigmák** és a **fejlesztői eszközök** területén. A feladat során a hallgatónak egy olyan alkalmazást kell készítenie, amelyben érvényesül a **procedurális**, **funkcionális** és **objektumorientált** szemlélet, miközben a rendszer képes valós adatok kezelésére, **aszinkron feldolgozásra**, **automatizált feladatvégzésre**, valamint az eredmények **vizualizálására** egy webes felületen.

Feladateleírás és követelmények

A hallgató készítsen egy **mikroszerviz-architektúrájú Python-alkalmazást**, amely legalább az alábbi komponenseket tartalmazza:

- **Backend szolgáltatás** FastAPI-val (REST API),
- **Frontend** Streamlit-tel (webes felület, <https://streamlit.io/>),
- **Adatbázis réteg** (pl. SQLite, PostgreSQL vagy más támogatott DB),
- **Automatizációs / háttérfolyamat** (időzített, aszinkron vagy web scraping feladat).

A rendszernek az alábbi technikai és tartalmi kritériumokat kell teljesítenie:

- **Virtuális környezetben** kell készülnie (`python -m venv venv`) és onnan futtathatónak kell lennie,
- **Önállóan futtatható** parancssorból (pl. backend indítása `uvicorn`-nal) és **Streamlit**-en keresztül (frontend futtatása),
- **Moduláris szerkezetű**, több `.py` fájlt tartalmaz, külön rétegekre bontva (pl. `models`, `services`, `api`, `frontend`),
- Tartalmaz **procedurális**, **funkcionális** és **objektumorientált** elemeket,

- A hallgató készítsen egy saját **FastAPI** alapú **REST backendet**, legalább néhány (pl. lista, részletek, statisztika) végponttal,
- A Streamlit alkalmazás **a saját FastAPI végpontjait hívja meg** és a kapott adatokat **Streamlit felületen jelenítse meg**,
- A backend az adatok tartós tárolásához **SQLAlchemy ORM**-et használ, és a bejövő/kimenő adatok szerkezetét **Pydantic modellek** határozzák meg,
- Tartalmaz legalább egy **API-hívás láncot**, ahol a backend külső vagy belső adatforrásból dolgozik (pl. fájl, adatbázis, külső API),
- Tartalmazzon **konfigurációkezelést** környezeti változókkal vagy **.env** fájljal (pl. API-kulcsok, adatbázis URL-ek biztonságos tárolásához),
- Képes **automatizált feladatvégzésre** (pl. időzített ütemezés, web scraping, automatikus adatfrissítés vagy e-mailküldés), például **threading**, **asyncio**, **schedule**, **smtplib**, **email.mime** vagy **BeautifulSoup** használatával.
- Tartalmaz **hibakezelést** és **logolást** (pl. **logging** modul),
- Tartalmaz legalább három **egységtesztet** **pytest**-tel, amelyek közül legalább egy a **@pytest.mark.parametrize** dekorátort használja,
- Tartalmazzon legalább egy **vizualizációt** (pl. diagram vagy statisztikai nézet Streamlit-ben),
- A kódfájlok elhelyezése legyen **logikus és átlátható** (minden modul a megfelelő mappában),
- Legyen egy **indítófájl vagy script** (pl. **main.py** vagy **start.sh**), amelyből a rendszer egyszerűen elindítható,
- Verziókövetéssel készül, **GitHub repository** formájában (<https://github.com/>),
- A projekt **Streamlit Cloudra** deployolható (<https://streamlit.io/cloud>), és ha a saját FastAPI backendnek interneten is elérhetőnek kell lennie, akkor azt **Render.com-on** kell futtatni (<https://render.com/>).

Értékelési szempontok

- **Kódminőség:** PEP8, docstringek, jól tagolt szerkezet, érthető névhasználat.
- **Paradigmák:** procedurális + funkcionális + OOP elemek tényleges és indokolt használata.
- **Architektúra:** backend–frontend–adatbázis–automatizáció komponensek tiszta szétválasztása, mikroszerviz-szemlélet.
- **Deploy és prezentáció:** működő Streamlit app, megfelelő dokumentáció, esetleges Dockerizálás **extra pontért**.

Leadás módja

A beadandó **GitHub repository** (<https://github.com/>) formájában adandó le, amely tartalmazza:

- a teljes forráskódot (.py fájlok) jól áttekinthető mappastruktúrában,
- requirements.txt a függőségekhez (pl. fastapi, uvicorn, streamlit, sqlalchemy, pydantic, requests, beautifulsoup4, python-dotenv, pytest, stb.),
- README.md, amely tartalmazza a projekt rövid bemutatását, az architektúra leírását, valamint a backend és frontend indításának módját (pl. uvicorn main:app -reload és streamlit run app.py),
- main.py vagy start.sh fájl a teljes rendszer indításához,
- a Streamlit deploy linkjét,
- ha a backend külön deployolva fut, annak **Render linkjét**.

A beadandót **be kell mutatni** személyesen vagy online, a működő projekt rövid ismertetésével. A bemutatás előtt **legalább 1 nappal** a GitHub repository linkjét és a deploy link(ek)et e-mailben el kell küldeni az oktatónak: kovacs.adam.eke@gmail.com.